

## A SYSTEM FOR GRAPHICALLY GENERATING ARABIC CHARACTERS

Syed S. Hyder and Henry K. Urion  
Université de Montréal

## ABSTRACT

The Arabic-Urdu-Farsi family of languages is used by over 300 million people. Unlike letters of the Roman alphabet, each Arabic character can assume different shapes which leads to a more compact representation of linguistic information, but which also causes increased difficulties in printing. In previous work [Hyder, 1971], a grammatical model was developed which determines the appropriate shape of a character as a function of the linking characteristics of adjacent letters during the generation of a word. As an outgrowth of this previous work, a system has been developed which graphically generates characters of the Arabic alphabet by using simple geometric primitives such as points, straight lines and arcs. Analysis has indicated that the various shapes of Arabic letters (over 100 in number) can be constructed from 35 subforms or primitives and 5 point configurations. A maximum of four primitives and one character dependent pattern of points are used to construct a character form. A context sensitive grammar has been developed for generating the graphic shapes and implemented on a CDC 1700/1747 digigraphic system. Some examples of character shapes generated by the grammar are given. The approach has possible applications in printing, photo type-setting and telecommunication of the Arabic-Farsi-Urdu family of languages.

## RESUME

Plus de 300 million de gens parlent une langue du type Arabe-Urdu-Farci. Cette étude a pour but de créer les caractères alphabétiques arabes à partir d'éléments géométriques simples: points, segments de droite et arcs. Cette recherche est la continuation de travaux précédents [Hyder, 1971]. Les caractères arabes, contrairement à l'alphabet latin, possèdent plusieurs formes qui, tout en facilitant la concision, offrent des difficultés d'écriture et de lecture. Nous avons développé dans des recherches antérieures un modèle grammatical qui détermine la forme d'un caractère suivant sa position dans un mot. Les analyses nous ont amené aux constatations suivantes: les différentes formes des caractères arabes (plus de 100) peuvent être construites à partir de seulement 35 sous-formes primitives et de 5 configurations de points. Chaque caractère a besoin, pour être défini, d'un maximum de 5 éléments primitifs et d'une configuration de points. Une grammaire répondant à ces critères a été créée et utilisée dans un système d'affichage graphique sur ordinateur 1700/1747. Des exemples de caractères générés sur écran cathodique sont donnés. Cette approche montre des applications possibles dans le domaine de l'imprimerie, de la composition optique et des télécommunications, ceci pour les langues du type Arabe-Urdu-Farci.

## 1. INTRODUCTION

The Arabic-Farsi-Urdu group of languages is used by over 300 million people living in North Africa, the Middle East, Iran, Pakistan, the Central Asian SSR Uzbekistan, Afganistan, Burma, Malaysia and Indonesia. This area is becoming increasingly important to the Western world in view of its political, economic and cultural potential. In these languages, the alphabetic characters have a close phonetic similarity to the corresponding characters of the English alphabet. However, the shape of a character is dependent upon its position in a word. The multiplicity of shape helps in information compression but leads to problems in writing because one must account for context dependency when choosing the form for a character as a word is formed.

A linguistic model, describing the manner in which character shapes are selected and linked with each other has been described earlier [Hyder, 1971]. During the course of that work, it became clear that the various graphic shapes corresponding to a single character, are composed of simple geometric forms: points, strokes and curves. The results of research to develop suitable combinational rules, which yield characters when primitive graphic shapes are synthesized, are presented.

It was discovered that the combinational rules constitute phrase structure grammars. The grammars have been simulated on the CDC-1700/1747 digigraphic system. Examples of the concatenation of graphic sub-forms according to the grammatical rules to form characters are given.

### 1.1 *Other research*

Little research has been devoted to the application of linguistic modeling to the generation of alphanumeric characters utilizing primitive geometric forms. Narasimhan [1966] first suggested a syntactic approach for resolving pattern recognition problems in picture processing, notably the analysis of bubble chamber photographs, and applied it to the generation of a subset of Roman letters. A phrase structure grammar was developed which is composed of a terminal vocabulary of eleven primitives, twelve non-terminal elements and thirty-one rewriting rules. It is a finite state grammar capable of producing nineteen letters of the Roman alphabet. Each of ten subforms, various straight and curved line segments, has a set of vertices associated with it. The vertices constitute the points where subparts concatenate. Any number of such points can be designated, but in Narasimhan's case, a maximum of only three vertices are stored as an integral part of any one subform. Each production rule specifies the following: 1) two elements of the vocabulary sets, either terminal or non-terminal; 2) which points of concatenation of the first subform link to those of the second; and 3) the vertices of the resulting phrase or linked primitives. Depending upon the complexity of the character, one to three production rules are applied during the generation of the letter. The grammar was programmed using a parallel processing simulator called COMPAX which was implemented on a CDC 3600 system. Output consisted of letters with subparts composed of series of X's.

Narasimhan's phrase structure grammar is straightforward but can be applied only to characters composed of primitives which directly interlink. Thus no discontinuities between subforms are permitted. The letter *t* for example, fulfills this condition whereas an *i* does not because of the separated dot. This aspect is regarded as the principal limitation of Narasimhan's work. Furthermore, it can be noted that undoubtedly little linguistic analysis of the primitive subforms was done in the process of deriving the production rules. Syntactic attributes of the subparts were apparently never considered as a means of reducing the number of production rules and simplifying the grammar.

In developing a transformational grammar to deal with the generation of Arabic characters, a more thorough analysis of the linking and syntactic attributes of each subform was conducted in order to define a concise transformational grammar with a minimum number of production rules. Furthermore, because characters such as *ك*, *س* and *ش* occur frequently in the Arabic alphabet, the issue of discontinuities between subparts could not be ignored. In many cases, as exemplified by the last two characters shown above, one can distinguish one character from another only by the disconnected point configuration affiliated with it. The following sections address these problems and explain how they are resolved.

## 2. DEVELOPMENT OF THE LINGUISTIC MODEL

### 2.1 *Analysis of Arabic characters*

A thorough study of the 96 distinctly different characters reveals that a maximum of only 35 fundamental subforms and five point configurations are needed to completely write any Arabic letter. The subforms  $\zeta_r$  for  $r = 1, 2, 3, \dots, 35$  and the null subform  $\epsilon$  constitute the vocabulary set  $V_S$ . The point configurations  $\pi_{uv}$  compose the vocabulary set  $V_P$ . The subscripts  $u$  and  $v$  indicate respectively the number of points in the configuration and the relative position of the arrangement of points either below or above the  $\zeta_r$  with which it is associated. They assume respectively the values 0, 1, 2 or 3 and 0 or 1.

As previously mentioned, one of the principal difficulties of composing characters using basic subforms centers on the specification of how the subforms and point configurations are correctly concatenated. In developing a scheme to cope with this problem, four kinds of operators  $\omega_t$  are used. The first operator  $\omega_1$  is simply the direct link. Given the string  $\zeta_{rp} \omega_1 \zeta_{sp'}$ ,  $\omega_1$  indicates that the end point of  $\zeta_{rp}$  is also the beginning point of  $\zeta_{sp'}$ , where the subscript  $r$  of  $s$  indicates a specific subform and  $p$  indicates the order in which the subforms are written, i.e.  $p' = p + 1$ . The second operator  $\omega_2$  is referred to as the point configuration interrupt. The string  $\zeta_{rp} \omega_2 \zeta_{sp'}$  is interpreted to mean that a point configuration  $\pi_{uv}$  must be written before  $\zeta_{sp'}$ . As in the case of  $\omega_1$ ,  $\zeta_{rp}$  links directly with  $\zeta_{sp'}$  after the points are written. The displacement of the arrangement of

points from the end of  $\zeta_{rp}$  is denoted by  $\beta(\zeta_r, \zeta_s, \pi_{uv})$ . The third operator  $\omega_3$  is the displacement interrupt which accounts for the set of strings  $\zeta_{rp} \omega_3 \zeta_{sp}'$ , in which the end point of  $\zeta_{rp}$  is not the beginning position of  $\zeta_{sp}'$ . Before  $\zeta_{sp}'$  can be written, a displacement  $\delta(\zeta_r, \zeta_s, \tau)$  where  $\tau$  indicates the type of Arabic character, i.e. isolated final, medial or initial ( $\tau = 1, 2, 3$  or 4 respectively) is performed. The character terminal displacement operator is  $\omega_4$ . It is needed in the case of isolated, final and some initial letters in order to properly adjust for the space which follows such characters. Given  $\zeta_{rp} \omega_4 \zeta_{sp}'$ , a terminal displacement  $\rho(\zeta_r, \zeta_s)$  occurs after  $\zeta_{rp}$ . In addition to these principal operators, there is one final null operator  $\omega_0$ . The operators,  $\omega_i$  where  $i = 0, 1, 2, 3$  or 4, form the vocabulary set  $V_0$ .

## 2.2 Transformational grammars

Having defined the variables and some of the notation, a 4-tuple phrase structure grammar  $G_\tau(V_T, V_N, P_\tau, \sigma_\tau)$  can be described for each of the four types of Arabic characters. These grammars were developed to succinctly represent the concatenation properties of the subforms and each consists of four parts. First, a terminal vocabulary is specified for each type  $\tau$ ,  $V_{T_\tau} = V_{S_\tau} \cup V_{O_\tau}$ . The non-terminal vocabulary  $V_N = \{O_1, O_2, O_3, O_4, O_5, \sigma_\tau\}$  where the  $O_t$  refers to an undefined operator and  $\sigma_\tau$  is the sentence symbol. The production rules  $P_\tau$  are of the form  $\phi \rightarrow \tau\psi$  where  $\phi$  and  $\psi$  can be phrases of mixed terminal and non-terminal elements, with  $\phi$  containing at least one non-terminal symbol. The grammars  $G_\tau$  are all of a context sensitive type. The non-terminal vocabulary is the same for each grammar while  $V_{T_\tau}$  and  $P_\tau$  are different for each type of Arabic character. The terminal vocabulary and production rules for the isolated, final, medial and initial characters are as follow.

### 2.3a Isolated characters

The terminal vocabulary for isolated characters  $V_{T_1}$  is composed of 21 subforms and all elements of  $V_0$ .

$$V_{S_1} = \{\zeta_1, \zeta_3, \zeta_4, \zeta_7, \zeta_8, \zeta_{10}, \zeta_{11}, \zeta_{13}, \zeta_{15}, \zeta_{16}, \zeta_{17}, \zeta_{18}, \zeta_{19}, \zeta_{20}, \zeta_{21}, \zeta_{22}, \zeta_{24}, \zeta_{25}, \zeta_{29}, \zeta_{31}, \zeta_{33}\}$$

$$V_{O_1} = \{\omega_0, \omega_1, \omega_2, \omega_3, \omega_4\}$$

$$V_{T_1} = V_{S_1} \cup V_{O_1}$$

The sentence  $\sigma_1$  is the general representation for all possible Arabic characters of the isolated form. The production rules which determine the operators in the phrase are:

$$I0: \sigma_1 \rightarrow O_1 \zeta_{i1} O_2 \zeta_{j2} O_3 \zeta_{k3} O_4 \zeta_{l4}$$

- I1:  $0_1 \zeta_{i1} 0_2 \zeta_{j2} \rightarrow \omega_3 \zeta_{i1} 0_2 \zeta_{j2}$  for  $\zeta_{i1} \in V_S$  and  $\zeta_{j2} \in V_S$  with displacement  $\delta(\zeta_i, \zeta_j, 1)$ .
- I2:  $\zeta_{rp} 0_p \zeta_{sp'} \rightarrow \zeta_{rp} \omega_4 \zeta_{sp'}$  for  $s = 0$  with displacement  $\rho(\zeta_r, \zeta_s)$ .
- I3:  $0_p \zeta_{rp} 0_p \zeta_{sp'} \rightarrow \omega_3 \zeta_{rp} 0_p \zeta_{sp'}$  for  $r = 22$  with displacement  $\delta(\zeta_{22}, \zeta_0, 1)$ .
- I4:  $0_1 \zeta_{i1} 0_2 \zeta_{j2} \rightarrow \omega_3 \zeta_{i1} 0_2 \zeta_{j2}$  for  $i \in 7, 17$  with displacement  $\delta(\zeta_i, \zeta_j, 1)$ .
- I5:  $0_p \zeta_{rp} 0_p \zeta_{sp'} \rightarrow \omega_2 \zeta_{rp} 0_p \zeta_{sp'}$  for  $p \in 1, 2$  and  $\pi_{uv}$  where  $u > 0$  and  $r \in 7, 10, 13, 15, 17, 18, 20, 33$  with point configuration displacement  $\beta(\zeta_r, \zeta_s, \pi_{uv})$ .
- I6:  $0_1 \zeta_{i1} 0_2 \zeta_{j2} \rightarrow \omega_2 \zeta_{i1} 0_2 \zeta_{j2}$  for  $i = 16$  and  $\pi_{uv}$  where  $u > 0$  with point configuration displacement  $\beta(\zeta_i, \zeta_j, \pi_{uv})$ .
- I7:  $\zeta_{i1} 0_2 \zeta_{j2} \rightarrow \zeta_{i1} \omega_2 \zeta_{j2}$  for  $i = 3$  and  $j = 4$  and  $\pi_{uv}$  where  $u > 0$  with point configuration displacement  $\beta(\zeta_i, \zeta_j, \pi_{uv})$ .
- I8:  $0_p \zeta_{rp} 0_p \zeta_{sp'} \rightarrow \omega_1 \zeta_{rp} 0_p \zeta_{sp'}$  for all other cases.

### 2.3b Final characters

Twenty-three subforms make up  $V_{T_2}$  and the complete set of operators are used.

$$V_{S_2} = \{\zeta_1, \zeta_2, \zeta_3, \zeta_4, \zeta_5, \zeta_7, \zeta_8, \zeta_9, \zeta_{10}, \zeta_{12}, \zeta_{13}, \zeta_{14}, \zeta_{15}, \zeta_{16}, \zeta_{17}, \zeta_{20}, \zeta_{21}, \zeta_{22}, \zeta_{24}, \zeta_{32}, \zeta_{33}, \zeta_{34}, \zeta_{35}\}$$

$$V_{O_2} = \{\omega_0, \omega_1, \omega_2, \omega_3, \omega_4\}$$

$$V_{T_2} = V_{S_2} \cup V_{O_2}$$

All possible Arabic characters of the final form can be expressed by the string  $\sigma_2$ . The following production rules yield the necessary specifications of the operators.

$$F0: \sigma_2 \rightarrow O_1 \zeta_{i1} O_2 \zeta_{j2} O_3 \zeta_{k3} O_4 \zeta_{l4} O_5 \zeta_{m5}$$

$$F1: O_2 \zeta_{j2} O_3 \zeta_{k3} \rightarrow \omega_3 \zeta_{j2} O_3 \zeta_{k3}$$

for  $j \in 1, 10$  with displacement  $\delta(\zeta_j, \zeta_k, 2)$ .

$$F2: \zeta_{rp} O_{p'} \zeta_{sp'} \rightarrow \zeta_{rp} \omega_4 \zeta_{sp'}$$

for  $s = 0$  with displacement  $\rho(\zeta_r, \zeta_s)$ .

$$F3: O_p \zeta_{rp} O_{p'} \zeta_{sp'} \rightarrow \omega_3 \zeta_{rp} O_{p'} \zeta_{sp'}$$

for  $r = 22$  and  $s = 0$  with displacement  $\delta(\zeta_{22}, \zeta_0, 2)$ .

$$F4: O_p \zeta_{rp} O_{p'} \zeta_{sp'} \rightarrow \omega_2 \zeta_{rp} O_{p'} \zeta_{sp'}$$

for  $p \in 1, 2, 3$  and  $\pi_{uv}$  where  $u > 0$  and  $r \in 10, 13, 15, 17, 19, 20, 33, 35$  with point configuration displacement  $\beta(\zeta_r, \zeta_s, \pi_{uv})$ .

$$F5: O_2 \zeta_{j2} O_3 \zeta_{k3} \rightarrow \omega_2 \zeta_{j2} O_3 \zeta_{k3}$$

for  $j \in 4, 16$  and  $\pi_{uv}$  where  $u > 0$  with point configuration displacement  $\beta(\zeta_j, \zeta_k, \pi_{uv})$ .

$$F6: O_1 \zeta_{i1} O_2 \zeta_{j2} \rightarrow \omega_2 \zeta_{i1} O_2 \zeta_{j2}$$

for  $i = 9$  and  $\pi_{uv}$  where  $u > 0$  and  $v > 0$  with point configuration displacement  $\beta(\zeta_i, \zeta_j, \pi_{uv})$ .

$$F7: O_2 \zeta_{j2} O_3 \zeta_{k3} \rightarrow \omega_2 \zeta_{j2} O_3 \zeta_{k3}$$

for  $j = 7$  and  $\pi_{uv}$  where  $u > 0$  and  $v = 0$  with point configuration displacement  $\beta(\zeta_7, \zeta_k, \pi_{uv})$ .

$$F8: \zeta_{rp} O_{p'} \zeta_{sp'} \rightarrow \zeta_{rp} \omega_4 \zeta_{sp'}$$

for  $s = 0$  with terminal displacement  $\rho(\zeta_r, \zeta_s)$ .

$$F9: O_p \zeta_{rp} O_{p'} \zeta_{sp'} \rightarrow \omega_1 \zeta_{rp} O_{p'} \zeta_{sp'}$$

for all other cases.

### 2.3c Medial characters

Only thirteen subforms are needed to create this type of character.

$$V_{S_3} = \{\zeta_1, \zeta_5, \zeta_6, \zeta_7, \zeta_9, \zeta_{11}, \zeta_{12}, \zeta_{15}, \zeta_{17}, \zeta_{19}, \zeta_{23}, \zeta_{27}, \zeta_{30}\}$$

$$V_{O_3} = \{\omega_0, \omega_1, \omega_2, \omega_3\}$$

$$V_{T_3} = V_{S_3} \cup V_{O_3}$$

Arabic letters of the medial form can be represented by the same general string used for final characters. The production rules follow.

$$M0: \sigma_3 \rightarrow 0_1 \zeta_{i1} 0_2 \zeta_{j2} 0_3 \zeta_{k3} 0_4 \zeta_{l4} 0_5$$

$$M1: 0_2 \zeta_{j2} 0_3 \zeta_{k3} \rightarrow \omega_3 \zeta_{j2} 0_3 \zeta_{k3}$$

for  $j \in 1, 23$  with interrupt displacement  $\delta(\zeta_j, \zeta_k, 3)$ .

$$M2: 0_p \zeta_{rp} 0_{p'} \zeta_{sp'} \rightarrow \omega_2 \zeta_{rp} 0_{p'} \zeta_{sp'}$$

for  $p \in 1, 2, 3$  and  $\pi_{uv}$  where  $u > 0$  and  $r \in 9, 15, 17, 19, 35$  with point configuration displacement  $\beta(\zeta_r, \zeta_s, \pi_{uv})$ .

$$M3: 0_2 \zeta_{j2} 0_3 \zeta_{k3} \rightarrow \omega_2 \zeta_{j2} 0_3 \zeta_{k3}$$

for  $j = 6$  and  $\pi_{uv}$  where  $u > 0$  with point configuration displacement  $\beta(\zeta_6, \zeta_k, \pi_{uv})$ .

$$M4: 0_p \zeta_{rp} \rightarrow \omega_0 \zeta_{rp}$$

for  $r = 0$ .

$$M5: 0_p \zeta_{rp} 0_{p'} \zeta_{sp'} \rightarrow \omega_1 \zeta_{rp} 0_{p'} \zeta_{sp'}$$

for all other cases.

### 2.3d Initial characters

These letters have a terminal vocabulary composed of thirteen subforms and all elements of  $V_0$ .

$$V_{S_4} = \{\zeta_1, \zeta_3, \zeta_6, \zeta_7, \zeta_{10}, \zeta_{11}, \zeta_{13}, \zeta_{15}, \zeta_{17}, \zeta_{18}, \zeta_{20}, \zeta_{23}, \zeta_{28}\}$$

$$V_{O_4} = \{\omega_0, \omega_1, \omega_2, \omega_3, \omega_4\}$$

$$V_{T_4} = V_{S_4} \cup V_{O_4}$$

The sentence  $\sigma_4$  represents all possible initial Arabic letter with the following production rules.

N0:  $\sigma_4 \rightarrow 0_1 \zeta_{i1} 0_2 \zeta_{j2} 0_3 \zeta_{k3} 0_4 \zeta_{l4}$

N1:  $0_1 \zeta_{i1} 0_2 \zeta_{j2} \rightarrow \omega_3 \zeta_{i1} 0_2 \zeta_{j2}$

for all  $\zeta_{i1} \in V_{S_4}$  and

$\zeta_{j1} \in V_{S_4}$  with interrupt displacement  $\delta(\zeta_i, \zeta_j, 4)$ .

N2:  $\zeta_{rp} 0_p \zeta_{sp}' \rightarrow \zeta_{rp} \omega_4 \zeta_{sp}'$

for  $r = 13$  with terminal displacement  $\rho(\zeta_r, \zeta_s)$ .

N3:  $\zeta_{i1} 0_2 \zeta_{j2} 0_3 \zeta_{k3} \rightarrow \zeta_{i1} \omega_2 \zeta_{j2} 0_3 \zeta_{k3}$

for  $i = 3$  and  $j = 13$  and  $\pi_{uv}$  where  $u > 0$  with point displacement  $\beta(\zeta_j, \zeta_k, \pi_{uv})$ .

N4:  $0_p \zeta_{rp} 0_p \zeta_{sp}' \rightarrow \omega_2 \zeta_{rp} 0_p \zeta_{sp}'$

for  $p \in 1, 2$  and  $\pi_{uv}$  where  $u > 0$  and  $r \in 7, 10, 13, 15, 17, 18, 20$  with point displacement  $\beta(\zeta_r, \zeta_s, \pi_{uv})$ .

N5:  $0_p \zeta_{rp} \rightarrow \omega_0 \zeta_{rp}$


for  $r = 0$ .

N6:  $0_p \zeta_{rp} 0_p \zeta_{sp}' \rightarrow \omega_1 \zeta_{rp} 0_p \zeta_{sp}'$

for all other cases.

2.4 PROCEDURE

A series of operators and subforms alternately positioned in a string constitute the input of the transformational grammars as previously defined. A forward scan of the string is performed which converts non-terminal specification of operators to terminal ones. In actual generation of a character, the subparts are written and linked in a right-to-left direction. The production rules are applied successively to a string which contains at least two subforms i.e.  $0_p \zeta_{rp} 0_p \zeta_{sp}'$  or  $\zeta_{rp} 0_p \zeta_{sp}'$  except in the case of initial characters where three subforms are considered. The use of a production rule determines which kind of operator is appropriate for the next undefined operator in the chain. The simultaneous consideration of two or three subforms is necessary because direct concatenation and discontinuities are determined by the syntactic attributes empirically affiliated with each primitive. The production rules concisely represent such linking characteristics.

Two examples will demonstrate the application of the rules. The character  is composed of the following subparts.

$\zeta_{i1} = \text{ـ} \quad i = 5$



$$\begin{aligned} \zeta_{j2} &= | & j &= 10 \\ \zeta_{k3} &= \smile & k &= 4 \\ \zeta_{\lambda 4} &= \surd & \lambda &= 22 \\ \zeta_{m5} &= \epsilon & m &= 0 \end{aligned}$$

$$\pi_{uv} = \text{null} \quad u = 0 \text{ and } v = 0$$

The particular rules used and how the input string is changed are given below at the left. The subforms which result from the application of production rules are shown at the right. A cross (X) is used to indicate the position after a displacement takes place before the following subform or configuration of points is displayed.

$$\sigma_{;2} \xrightarrow[\text{F0}]{\text{Rule}} 0_1 \zeta_{5,1}^* 0_2 \zeta_{1,2} 0_3 \zeta_{4,3} 0_4 \zeta_{22,4} 0_5 \zeta_{0,5}$$

$$0_1 \zeta_{5,1} 0_2 \zeta_{1,2} \xrightarrow[\text{F9}]{\text{Rule}} \omega_1 \zeta_{5,1} 0_2 \zeta_{1,2}$$

$$\omega_1 \zeta_{5,1} 0_2 \zeta_{1,2} 0_3 \zeta_{4,3} \xrightarrow[\text{F1}]{\text{Rule}} \omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} 0_3 \zeta_{4,3}$$

with displacement  $\delta(\zeta_1, \zeta_4, 2)$

$$\omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} 0_3 \zeta_{4,3} 0_4 \zeta_{22,4} \xrightarrow[\text{F9}]{\text{Rule}} \omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} \omega_1 \zeta_{4,3} 0_4 \zeta_{22,4}$$

$$\omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} \omega_1 \zeta_{4,3} 0_4 \zeta_{22,4} 0_5 \zeta_{0,5} \xrightarrow[\text{F3}]{\text{Rule}}$$

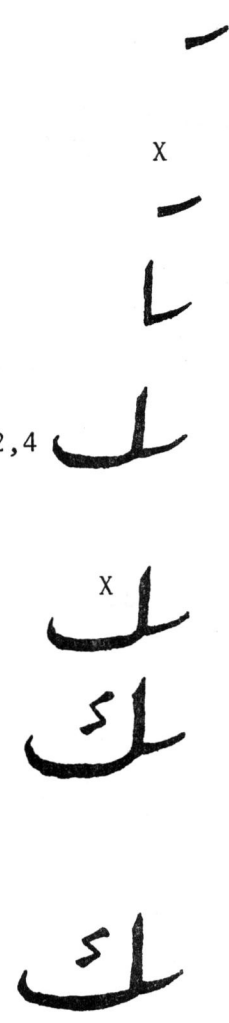
$$\omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} \omega_1 \zeta_{4,3} \omega_3 \zeta_{22,4} 0_5 \zeta_{0,5}$$

with displacement  $\delta(\zeta_{22}, \zeta_0, 2)$


$$\omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} \omega_1 \zeta_{4,3} \omega_3 \zeta_{22,4} 0_5 \zeta_{0,5} \xrightarrow[\text{F8}]{\text{Rule}}$$

$$\omega_1 \zeta_{5,1} \omega_3 \zeta_{1,2} \omega_1 \zeta_{4,3} \omega_3 \zeta_{22,4} \omega_4 \zeta_{0,5}$$

with displacement  $\rho(\zeta_{22}, \zeta_0)$



\* For the sake of clarity, a comma has been inserted between the two numerical subscripts of the  $\zeta$ 's.

As a second example, the procedure used to generate the character  is indicated below. It exemplifies the way in which a configuration of points are "linked" to a subform. The following subparts are used.

$$\zeta_{i1} = \text{ـ} \quad i = 12$$

$$\zeta_{j2} = \text{و} \quad j = 15$$

$$\zeta_{k3} = \text{و} \quad k = 16$$

$$\zeta_{\ell 4} = \varepsilon \quad \ell = 0$$

$$\pi_{uv} = \text{•} \quad u = 3 \text{ and } v = 1$$

$$\sigma_2 \xrightarrow{\text{Rule F0}} 0_1 \zeta_{12,1} 0_2 \zeta_{15,2} 0_3 \zeta_{16,3} 0_4 \zeta_{0,4}$$

$$0_1 \zeta_{12,1} 0_2 \zeta_{15,2} \xrightarrow{\text{Rule F9}} \omega_1 \zeta_{12,1} 0_2 \zeta_{15,2}$$

$$\omega_1 \zeta_{12,1} 0_2 \zeta_{15,2} 0_3 \zeta_{16,3} \xrightarrow{\text{Rule F4}} \omega_1 \zeta_{12,1} \omega_2 \zeta_{15,2} 0_3 \zeta_{16,3}$$

with displacement  $\beta(\zeta_{15}, \zeta_{16}, \pi_{3,1})$

$$\omega_1 \zeta_{12,1} \omega_2 \zeta_{15,2} 0_3 \zeta_{16,3} 0_4 \zeta_{0,4} \xrightarrow{\text{Rule F9}}$$

$$\omega_1 \zeta_{12,1} \omega_2 \zeta_{15,2} \omega_1 \zeta_{16,3} 0_4 \zeta_{0,4}$$

$$\omega_1 \zeta_{12,1} \omega_2 \zeta_{15,2} \omega_1 \zeta_{16,3} 0_4 \zeta_{0,4} \xrightarrow{\text{Rule F8}}$$

$$\omega_1 \zeta_{12,1} \omega_2 \zeta_{15,2} \omega_1 \zeta_{16,3} \omega_4 \zeta_{0,4}$$

ـ  
و  
و  
ش

ش

ش

### 3. IMPLEMENTATION

The set of transformational grammars described above have been verified by computer simulation as a method of generating Arabic characters utilizing subforms. The system has been implemented on a CDC 1700/1747 digigraphic computer at the Université de Montréal. Some aspects of the programming and sample outputs are given in this section.

The thirty-five subforms,  $\zeta_i$ , and five configurations of points,  $\pi_{uv}$ , have been created on a 274 CRT<sup>r</sup> screen using a light pen. Each primitive is graphically defined as a straight or curved line segment of varying thickness with the beginning and ending points specified. The configuration of points are defined in a similar manner. Each subpart is labeled and stored on disc as a graphic macro, the average size of which is approximately ten bytes. Various horizontal and vertical displacement macros are also defined which are used when interrupt, point and terminal displacements are needed.

The user specifies a certain Arabic letter  $C_i$  where  $i$  indicates the form of the character (figure 1). The subforms  $\zeta_i$  needed to generate the character are retrieved from memory and necessary information concerning the configuration of points is determined. Depending on the form of the letter selected, one of the four routines simulating the transformational grammars is used. To control the rate at which the program "builds" a character with the subforms, the user specifies when he wishes to have an additional subpart displayed on the screen. In such a way, one is able to take pictures of the gradual formation of a character as each primitive is displayed. Some examples of Arabic characters generated in this manner by the program are given in figure 2.

#### 4. CONCLUSION

A computer simulation for graphically creating Arabic characters validates the transformational grammars developed to determine the concatenation operators according to the context of the elements in the sentences  $\sigma_T$ . A mechanism for generating alphanumeric character composed of linked and disconnected subparts has been developed. This procedure for generating characters depends, nevertheless, on stored information which indicates the specific subforms used in each Arabic letter and the displacements needed between certain subparts. Additional research is being conducted with the goal of refining the model whereby linguistic information, explicitly defined at present, can be compressed into additional sets of rules thus further simplifying the procedure. A hardware implementation of the character generator is an additional possibility and its feasibility is also under consideration. The approach could have considerable applicability in the the fields of photo type-setting and telecommunications in parts of the world where the Arabic family of languages is used.

#### ACKNOWLEDGEMENTS

The work has been supported by the National Research Council of Canada grant A-2164 and the International Development Research Center. The authors are grateful to Dr. M. Gold for his suggestions and comments, and thankfully acknowledge the programming assistance provided by Daniel Osenda and André Fourrier.

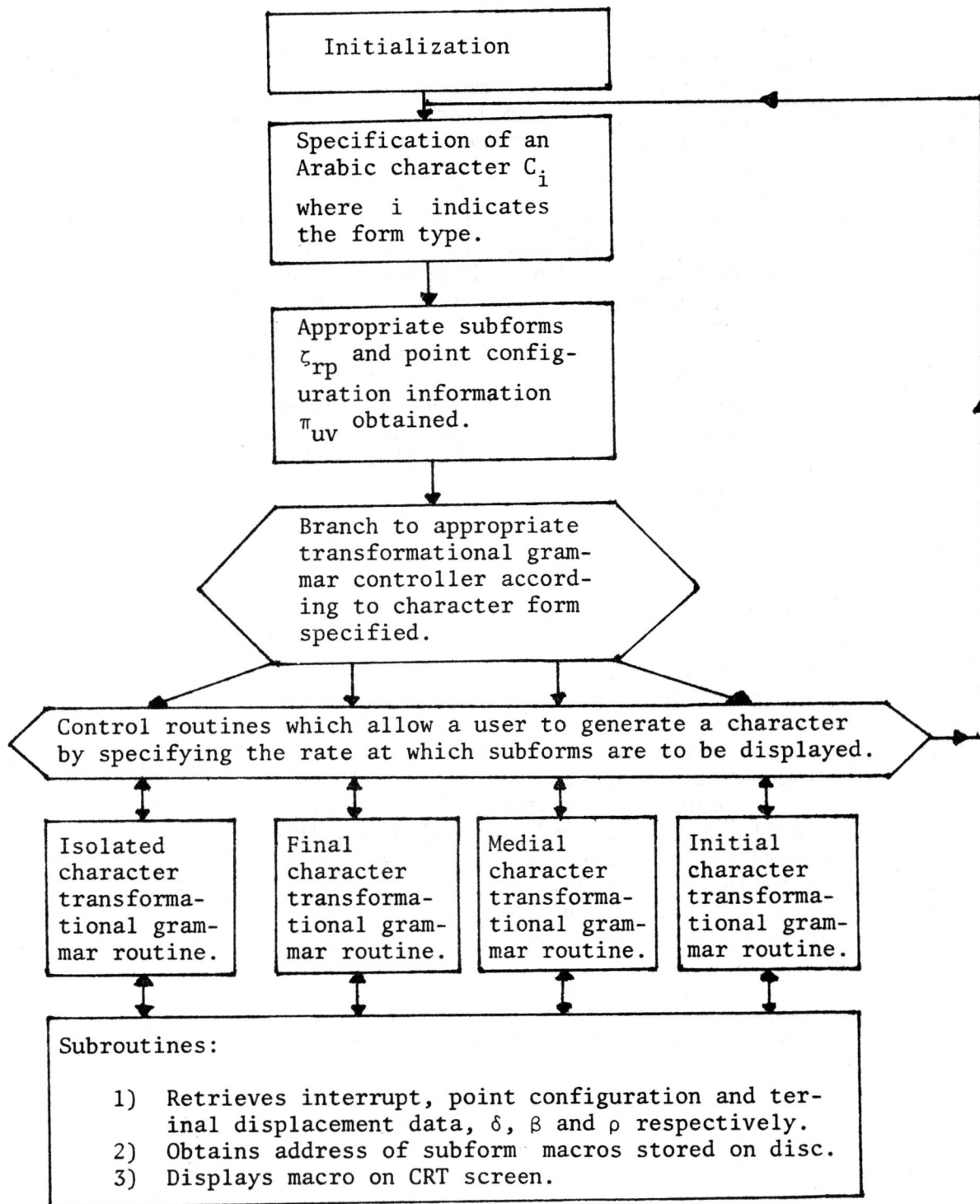
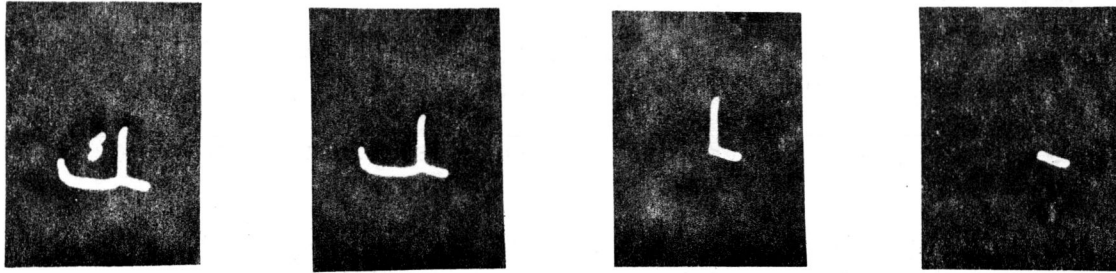
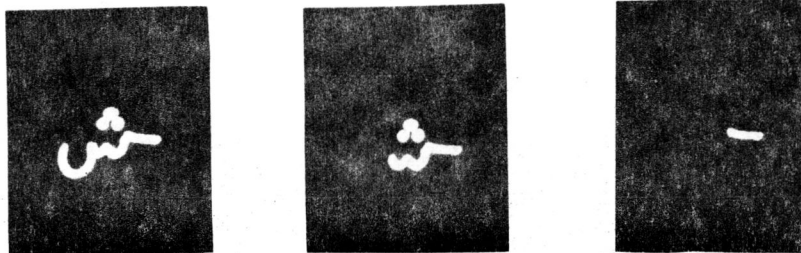


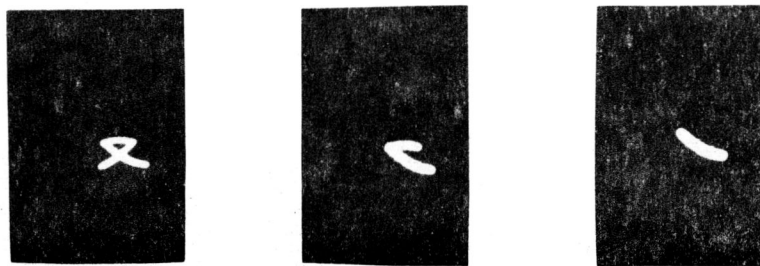
Figure 1. Block diagram of Arabic character graphical generator.



2a. Medial (k)



2b. Final (sh)



2c. Medial (e)

Figure 2. Sample output of Arabic characters on a 1700/1747 digigraphic system utilising subforms.

## REFERENCES

- [1] Hopcroft, J. E. and Ullman, J. E. Formal Languages and Their Relations to Automata, Addison Wesley, 1969.
- [2] Naraximhan, R. "Syntax-Directed Interpretation of Classes of Pictures", Comm. ACM, 9(1966), p. 166-173.
- [3] Hyder, S. S. "A System for Generating Urdu/Farsi/Arabic Script", Proceedings of IFIP Congress, 1971.
- [4] Meynet, R. "L'écriture Arabe en question", Publication du Centre Culturel Universitaire, Dar-el-Machreq, Beyrouth, Liban.